LA-UR-19-26385
Approved for public release; distribution is unlimited.

Title:	Lecture 2: Basic Code Usage
Author(s):	Grove, John W.
Intended for:	Report
Issued:	2019-07-08

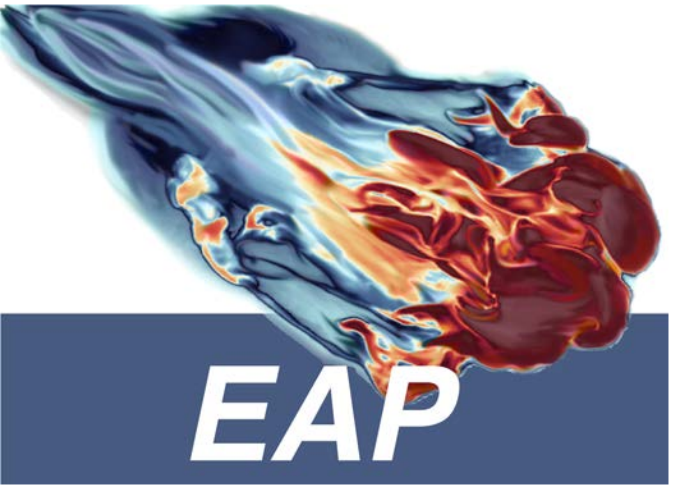
Los Alamos
NATIONAL LABORATORY
EST. 1943

Los Alamos
NATIONAL LABORATORY
EST. 1943

Delivering science and technology
to protect our nation
and promote world stability

National Nuclear Security Administration

Lecture 2: Basic Code Usage



John W. Grove

CCS-2

June 6, 2019

Getting Started to Use xRage?

Where to Get Help

- Like all complicated projects, using xRage has a big learning curve
- We try to make things as simple as possible but what is obvious to an experienced user may not be so obvious to new users
- Your first line of help is to send email to
  - crestone_support@lanl.gov
    - Use this email for general questions about the code
  - consult@lanl.gov
    - This is the email to the lab HPC consultants. This address is useful for machine problems including down time due to scheduled maintenance.
    - You can also call 5-4444 to get help
    - Tickets related to system problem can also be created via the AskIT web site.
- User guides for the codes can be found on the yellow at /usr/projects/eap/doc
  - xrage_userman.pdf is the guide for xRage

xRage Executable

- The xRage executable is a UNIX command line executable and is run in a variety of ways
  - Direct command line entry, usually with the appropriate MPI front end such as mpiexec or mpirun
  - Indirectly via batch processing scripts
    - These will be discussed in more detail later
  - Indirectly via other software tools
- The output from xRage consists of a variety of files
  - Full restart dumps in binary format
  - Diagnostic output, usually in ASCII
    - These include things like tracers
  - Graphics output including
    - Ensight
    - Paraview
    - HDF (4 or 5)
    - Tektronix output (yes it’s old, but still works with xterm in vt100 mode)
  - Log files showing run progress and warning and error messages.
- Currently the code is best supported for Linux based LANL HPC systems
  - Other platforms, such as standalone Linux systems or MacOS require a bit of effort to make the code work.

Where are the Executables Located?

- Running an executable requires first knowing where it lives or how to create it
  - Building the code will be discussed later
- Precompiled executables
  - On the yellow network these can be found in the directory /usr/projects/eap/releases
  - Requires membership in the dacodes group
- The symbolic link “latest” points to the most recent release
- The subdirectory xrage will contain the code executables
  - The naming convention is xrage.*N*.YYYYMMDD[_CompiledOption]_MACHINE.x
  - *N* will usually be 1
  - The date is given by year month day, such as 20190220.
  - CompiledOption indicates how the code was built
    - relq_debug uses the release compile options with debugging
    - testq_debug_serial is a serial compiled version with debugging
  - MACHINE is currently one of the following
    - SN a tri-lab commodity version portable to most of the HPC platforms
    - TR is for the Haswell partition on trinitite
    - TR_KNL uses the Knights Landing platform on trinitie
- Generally you most likely want either xrage.1.YYYYMMDD_SN.x or xrage.1.YYYYMMDD_TR.x

Running xRage

- LANL HPC resources for running xrage include:
  - Snow – ssh sn-fey
  - Grizzly – ssh gr-fey
  - Trinitite – ssh tt-fey
- Obviously you must have access to one or more of these or similar resources
  - Visit register.lanl.gov to request access to these and other resources
- Once you are logged into an appropriate machine the next slide describes how to set your UNIX environment to run the code

UNIX Environment Setup

- It is strongly recommended that you use the provided environment files before running the code
  - Csh – source /usr/projects/eap/dotfiles/.cshrc
  - Bash – source /usr/projects/eap/dotfiles/.bashrc
- If you plan to run the code frequently, you might want to add these lines to your system .cshrc, .profile, or .bashrc file as appropriate
- If you have checkout access to the git repository you can use the corresponding versions of these files in your checkout
  - eap.xrage/tools.xrage/Environment/.cshrc or .bashrc as appropriate
- These setup files load modules and set environment variables needed to compile and use the code
- If you run into trouble send email to crestone_support@lanl.gov

Getting an Execution Node

- The HPC computers are set up as front/back end machines. You log into the front end and must run executables on the backend.
- The tools to access the back end nodes and to run programs are part of the SLURM package
  - https://hpc.lanl.gov/slurm_introduction
  - https://slurm.schedmd.com/overview.html
- You don't need to know everything about SLURM
- The EAP project provides a number of tools that make using SLURM easier e.g.
  - Interactive login – the eap environment contains a simple alias myllogin to get interactive sessions "myllogin –n 2" gives you a two MPI rank allocation
  - Batch job submission – the Perl script run_job.pl is a handy tool provided by EAP for running larger batch processes
- The main tool to access backend resources is salloc, see the unix manual page for more details

Running xRage?

Example: Running xRage

- We will run a simple one-dimensional Riemann Problem solution
  - Specifically the Sod 1D shock tube: Sod, G. (1978). "A Survey of Several Finite Difference Methods for Systems of Nonlinear Hyperbolic Conservation Laws." J. Comp. Phys. **27**: 1.
- From my workstation I execute
  - kinit –f
    - Get a Kerberos Ticket
  - ssh –Y sn-fey
    - Log into Snow
  - myllogin –n 1 or salloc --N 1 --t 4:0:0 --qos=interactive
    - Get a backend allocation to run the code
  - mpirun –np 1 /usr/projects/eap/releases/latest/xrage/xrage.1.190218_SN.x sod.input >& sod.output &
- This runs xrage using the input file to be described shortly.

Advanced Run Options

- Although the use of batch processing is overkill for this simple example, non-trivial applications will benefit from the use of the automated run scripts provided by M. McKay.
- These scripts provide a variety of convenient options for users
  - Automatic back end allocation and job submission
  - Automatic restart and job chaining
    - When you approach the maximum time for a backend allocation the code will create a restart dump and exit gracefully
    - A new submission is automatically created that will restart your run and continue as soon as a new backend allocation is available.
    - When the run is stopped a cleanup can be performed to organize the output into appropriate locations.
  - In many cases your application will continue until the appropriate termination criteria is reached (such as maximum simulation time).
    - Runs can go for weeks or even months with little user intervention needed
    - When the system is brought down gracefully, such as for dedicated system time, your application will automatically restart
    - In the event of abnormal system stoppages it may be necessary to manually restart an applications, but the provided scripts make this easy.

Advanced Run Options

- The run_job.pl script
  - A perl script to create and submit batch jobs on HPC machines
- Path: /usr/projects/eap/tools/run_job.pl
  - If you have checked out the source code this script can also be found in eap.xrage/tools.xrage/General/run_job.pl
- Basic usage
  - To create a run script for a specific application go to the directory where you want the output to appear and execute the interactive command
  - run_job.pl --convert basic
    - You will be prompted for a variety of options in most cases the defaults are what you should use
      - Exceptions include the use of a specific xRage executable, the size of the requested allocation, and the locations of input files
    - After completion you will find a c-shell script run_job.csh has been created in the directory where you ran run_job.pl --convert basic.
    - This script can be edited and modified

Advanced Run Options

- An example run_job.csh script:
- You may want to change the value for RJ NUMPE to increase or decrease the number of MPI ranks used to run the code.
  - If you do this for a running job the change will become effective at the next restart
- Once the run_job.csh script is created you start your job with the command “run_job.pl go”
- More options can be found by the command run_job.pl --help
- Mostly this script is portable across HPC platforms except for SLUM specific arguments like the quality of service request --qos

```
#!/bin/tcsh -f
#RJ BATCH              = yes
#RJ BATCH_ARGS         =  --qos=dev
#RJ CHAIN              = yes
#RJ EXEC               = myexec
#RJ GROUP              = eapdev
#RJ NUMPE              = 8
#RJ PNAME              = RJEXAMPLE
#RJ TIME               = 8:00:00
#RJ UMASK              = 27

# cleanup from previous run
run_job_cleanup.pl

# teos_out generation
if ( -e "myteos.in" && ! -e "myteos_out" ) then
  &&&RJ_CMD_PRUN&&& myteos.in
endif

# run
&&&RJ_CMD_PRUN&&& myinput.in -v secmax=$RJ_TIME_REMAINING_B

# cleanup and check for continue
run_job_cleanup.pl --check
```

sod.input

- This is the text I used for the xRage input sod.input

```
pname = "sod"

kread = -1        ! this value creates a new run as compared to a restarted run
ncmax = -1        ! no maximum cycle number, run until tmax is reached
tmax = 0.01       ! run until simulation time tmax [seconds]
tedit = 0.01      ! output a restart dump file every tedit [seconds]

imxset = 100       ! 100 zones starting at x = 0
dxset = 20         ! zone size of 20 [cm], domain runs from 0 to imxset*dxset = 2000

! MATERIALS
keos = 0                 ! Perfect gas equation of state
nummat = 1               ! Only one EOS model
matdef(16,1) = 0.4       ! Gruneisen export, perfect gas gamma = 1.4
matdef(30,1) = 1.0e11    ! CV, specific heat at constant volume, ergs/(gm eV/k) eV = electron volts,
                         ! k = Boltzmann constant computed so that P0/rho0 = (gamma-1)*CV*T0 T0 = 0.025 eV/k = 290.11298 K

! REGIONS
numreg = 2              ! Two regions to initialize

matreg(1) = 1           ! Region one is the whole domain by default
rhoreg(1) = 1.25e-4     ! Mass Density [grams/cc]
siereg(1) = 2.0e9       ! Specific internal energy [erg/gram] pressure = (gamma-1)*rho*sie = .1 bar = 1e5 mubar

matreg(2) = 1           ! Region two will overwrite region one for xlreg(2)<x<xrreg(2)
xlreg(2) = 0.0          ! Region two starts at xlreg(2) = 0.0
xrreg(2) = 1000          ! Region two ends at xrreg(2) = 1000, Initial location of the discontinuity [cm]
rhoreg(2) = 1.0e-3      ! Mass Density [grams/cc]
siereg(2) = 2.5e9       ! Specific internal energy [erg/gram] pressure = (gamma-1)*rho*sie = 1 bar = 1e6 mubar
```

- This is about as a “bare bone” input file as I could create containing almost the minimum needed to create a runnable input file
- Specification of the Sod test problem can be found at http://www.thevisualroom.com/sods_test_problem.html
  – Our formulation is slightly modified to run from zero to 20 meters instead of -10 to 10 meters

sod.input explained

- The lines in this file direct the code to behave in the following ways
- A full description of input file construction is beyond the scope of this course
  - See the xRage users guide for more complete details
    - /usr/projects/eap/doc/xrage_userman.pdf
  - The structure of the input file is of the form *variable*=*value* where variable is usually an internal xRage variable whose value affects the initialization or running of the code. These variables can real, integer, logical, or character valued, and may be arrays.
- Comments begin with an exclamation mark (!)
  - These can occur anywhere in the input file
  - Any text following a ! is ignored by the input file reader
  - Use of comments can greatly increase ones ability to understand an input file and are encouraged for good style
    - After a week you may forget what an input line meant
    - Sooner as you get older 🙂

sod.input Name the problem

- pname = ‘sod’
- pname is the character valued variable used to build the strings used for output file names.
- The default value is the base name for the executable.
- The output files are named using the value of pname
  - Examples
    - sod-dmp000000: name of the first dump file, subsequent dumps are named using the time cycle values
    - sod-output: contains information about the way the run was executed
- Generally you should give this variable a value that makes sense to you.

sod.input restart

- kread = -1
- This variable determines whether the run is a new run or a restart
- The default value of 0 would try to restart from a dump at cycle 0
- The value -1 indicates that this is a new run
- We will go into much more detail on how restart works shortly
- This example does not use restart so kread is the only value needed

sod.input output control

- ncmax = -1
  - The value ncmax is the maximum number of cycles to be computed
  - The value -1 indicates that no maximum number of cycles is requested and the simulation will continue until some other stop criteria is reached.
- tmax = 0.01
  - tmax is the maximum simulation time for the run. When the accumulated simulation time equals or exceeds this value the run will stop
  - Here the run stops with the simulation time reaches 0.01 seconds
- tedit = 0.01
  - tedit tells the code how often to print restart dumps, in this case every 0.01 simulation time seconds. Here, this will produce two restart dumps one at time zero and the second after time 0.01. The exact time of the last dump depends on the size of the time step. By default the code will not try to adjust the time step to satisfy the editing options.

sod.input mesh specification

- imxset = 100
  - Create an initial level 1 mesh with 100 cells in the x-direction
  - 2D runs would also specify jmxset
  - 3D runs specify all of imxset, jmxset, and kmxset
- dxset = 20
  - Sets the mesh size for the level 1 mesh at 20 cm
  - 2D would also specify dyset
  - 3D specifies dxset, dyset, and dzset
- By default the computational domain starts at the coordinate value of zero and extends to the value imxset*dxset
  - So the domain here is $0 \leq x \leq 2000$
  - For 2D and 3D the domains are set in a similar fashion
- You can change the origin by setting values for xzero, yzero, or zzero as appropriate

sod.input equation of state

- keos = 0
  - Defines the equation of state to be used as the perfect gas equation of state
    - $e = \frac{PV}{(\gamma - 1)} = C_V T$
- nummat = 1
  - The value states the number of materials in the problem, in this case one material
  - Different materials will generally use different equations of states
- matdef(16,1) = 0.4
  - Defines the Grüneisen exponent $\gamma - 1$
  - this value is dimensionless
- matdef(30,1) = 1.0e11
  - Defines the value for the specific heat at constant volume
  - units of (erg/gram/(eV/k)
  - k is the Boltzmann constant to convert energy units of electron volts (eV) into temperature units.
  - 1 eV/k = 11604.519 degrees Kelvin
  - It is common for users to simply state temperatures in electron volts but what they really mean is electron volts per Boltzmann constant

sod.input region specification

- numreg = 2
  - Two initial regions will be specified
- matreg(1) = 1
  - sets the material for region 1 to to material 1
  - region 1 is special in the sense that it is generally the whole computational domain
  - subsequent regions will overwrite the values for preceding regions
- rhoreg(1) = 1.24e-4
  - Sets the density of region 1 to 1.25×10^{-4} grams/cc
- siereg(1) = 2.0e9
  - Sets the specific internal energy of region 1 to 2.0×10^{9} ergs/gram
- matreg(2) = 1
  - Region 2 contains material 1
- xlreg(2) = 0.0
  - region 2 starts at x = 0.0
- xrreg(2) = 1000
  - region 2 ends at x = 1000
- rhoreg(2) = 1.0e-3
  - Sets the density of region 2 to 1.0×10^{-3} grams/cc
- siereg(2) = 2.5e9
  - Sets the specific internal energy of region 2 to 2.5×10^{9} ergs/gram

Visualization

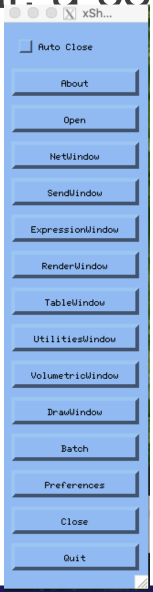
- We have a computed solution. How do we examine it?
- Several tools are available to visualize xRage solutions
- Ensight - https://www.ansys.com/products/fluids/ansys-ensight
  - This is a commercial product.
  - Available on HPC systems
  - On HPC systems the EAP environment setup will set up this tool and add its executable locations to your path
  - Workstation copies can be installed. Contact Robert Kares rjk@lanl.gov for information on the installation
- Paraview - https://www.paraview.org/
  - Open source tool with similar capabilities to Ensight
  - Available via modules on HPC systems
- xshow - /usr/projects/eap/tools/show/Linux/xshow
  - SAIC developed X11 gui tool for visualizing xRage
  - Very popular with users

Visualization – Other tools

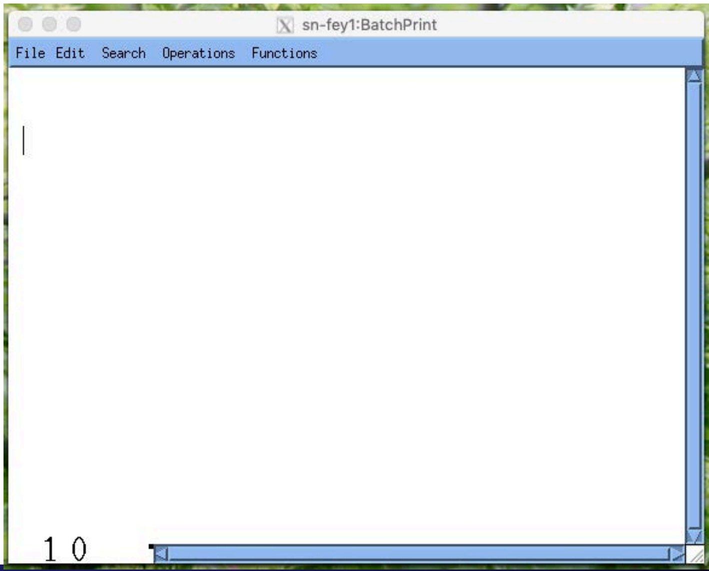
- Many other tools have been used to analyze xRage simulations
- Often these may require some programming to use and may require input file options to print files suitable for use by these tools
  - Actually, this point applies to both Ensight and Paraview as well
- Examples include
  - gnuplot - http://www.gnuplot.info/
  - pythonplot - Python based tool for plotting
  - Matlab - https://www.mathworks.com/
    - Commercial tool requires license
  - Mathematica - https://www.wolfram.com/mathematica/
    - Commercial tool requires license, available via ESD at LANL
- Other tools for examining image files include
  - ImageMagick - http://www.imagemagick.org/
    - Can display and convert a variety of graphics formats
- There are many other tools. The key is to find a tool that does what you need to do

Example: Using xshow

- The xshow tool is by far the most popular tool for visualizing xRage solutions.
- What follows is an example of how to use xshow to examine the restart dumps that were created with the Sod problem example
- Start xshow – assuming that you are using the EAP environment as described previously the executable should be in your path and all you need to do is run the command “xshow”
- Two windows appear, a command window and a graphic window
  - Command window

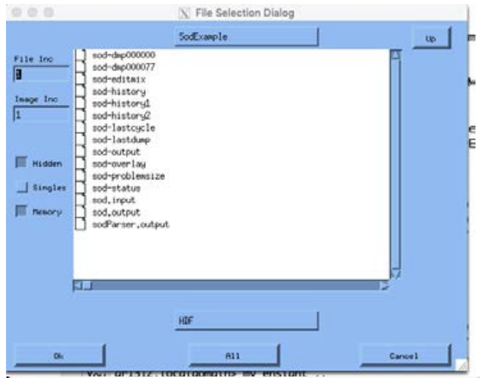


Graphics window

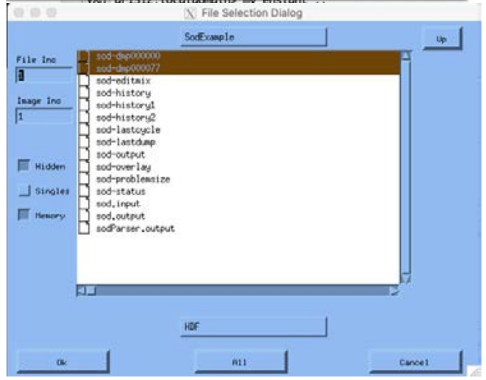


Example: Using xshow

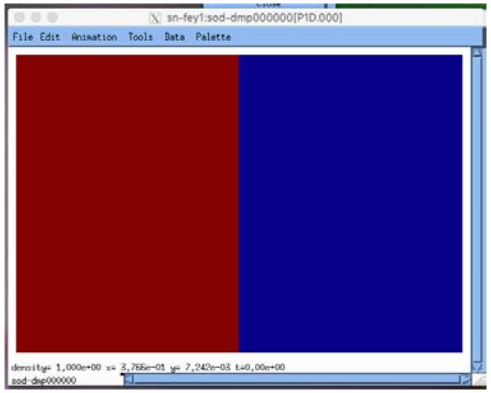
- Open the dump files
  – On the command window select open
- Highlight the files you want to open
  – Select OK
  – We used the default file type of HDF
- The result is a 2D plot of the 1D solution
  – Values on the vertical axis are constant here
- The next slide shows how to obtain a 1D plot of the solution density



```
sod-dmp000000
sod-dmp000077
sod-editmix
sod-history
sod-history1
sod-history2
sod-lastcycle
sod-lastdump
sod-output
sod-overlay
sod-problemsize
sod-status
sod.input
sod.output
sodParser.output
```



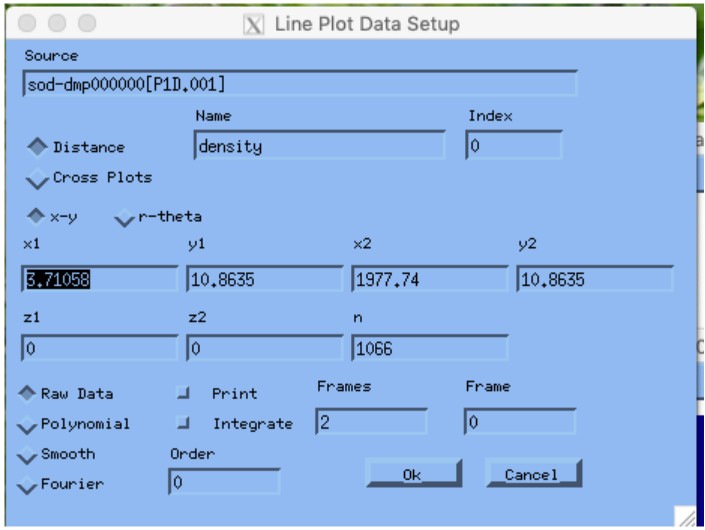
```
sod-dmp000000
sod-dmp000077
sod-editmix
sod-history
sod-history1
sod-history2
sod-lastcycle
sod-lastdump
sod-output
sod-overlay
sod-problemsize
sod-status
sod.input
sod.output
sodParser.output
```


sn-fey1:sod-dmp000000[P1D.000]
File Edit Animation Tools Data Palette

```
density= 1.000e+00 x= 3.766e-01 y= 7.242e-03 t=0.00e+00
sod-dmp000000
```

Example: Using xshow

- To plot density we will take a horizontal slice of the image obtained on the last step
- Under the Tools option select Line Plots
- Drag a line from left to right across the color image
  - Adjust the line until it looks horizontal and straight
- The Line Plot Data Setup window will appear
- The variable density is selected by default you can change this to other variables as appropriate
- By default the graph will be density verses distance
- Select OK
- A new window will appear showing density verses distance for the first dump in the list


Line Plot Data Setup

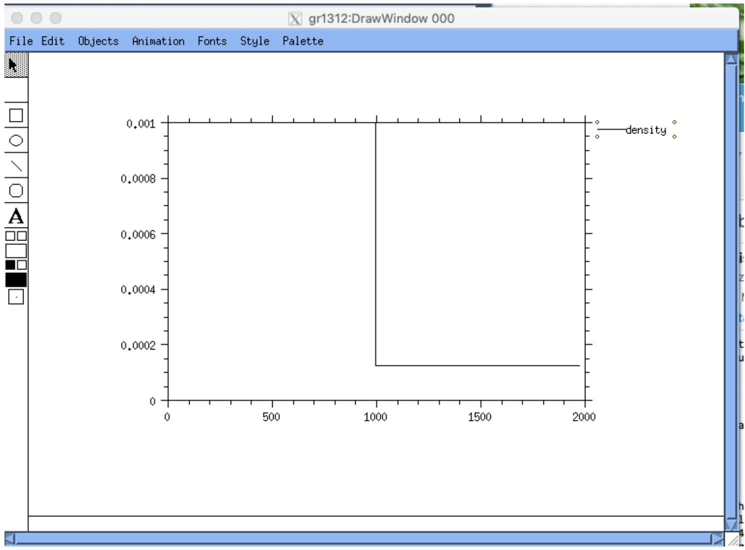
Source: sod-dmp000000[P1D.001]

Distance — Name: density — Index: 0
Cross Plots
x-y   r-theta

x1	y1	x2	y2
3.71058	10.8635	1977.74	10.8635

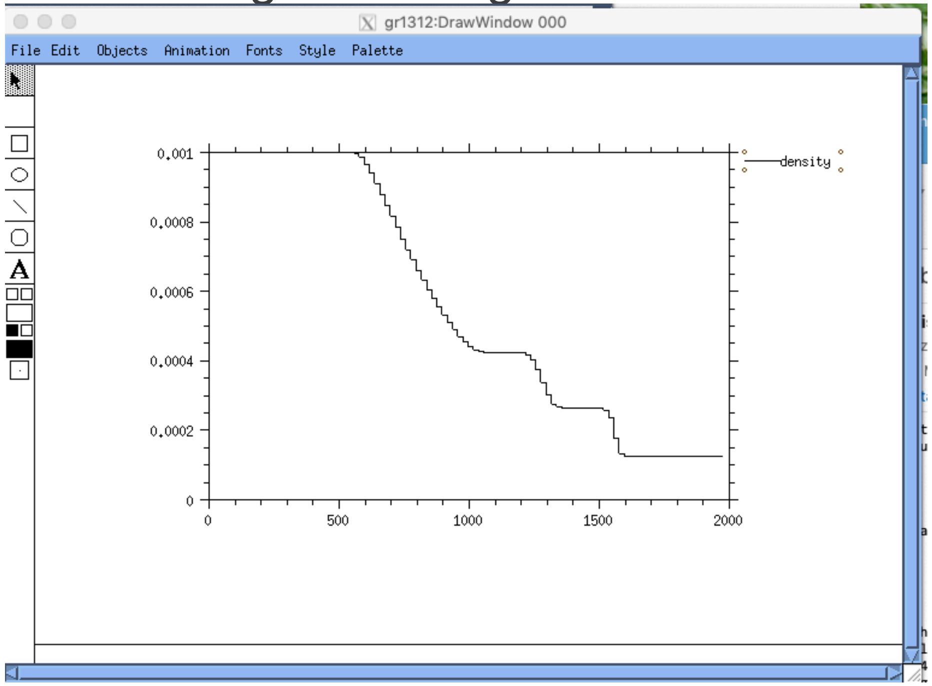
z1	z2	n
0	0	1066

Raw Data   Print   Frames: 2   Frame: 0
Polynomial   Integrate
Smooth   Order: 0
Fourier



Example: Using xshow

- You can move forward and backwards among the frames using the Animation option.
- Select Forward the the solution will move through the images until it reaches the last frame
- Other options allow you to step one at a time through the images move forward and backwards or start a looping animation
- More advanced usage allow you to view multiple plots in separate windows and synchronize the animation between the separate windows
- xshow is a powerful tool and this example just barely touches its full capabilities



- Unfortunately there is not much documentation for xshow, but the GUI is fairly intuitive. Ask colleagues or send email to crestone_support@lanl.gov for assistance in more advanced usage.
- To end the xshow session select Quit from the file menu.

Example: Using Ensight

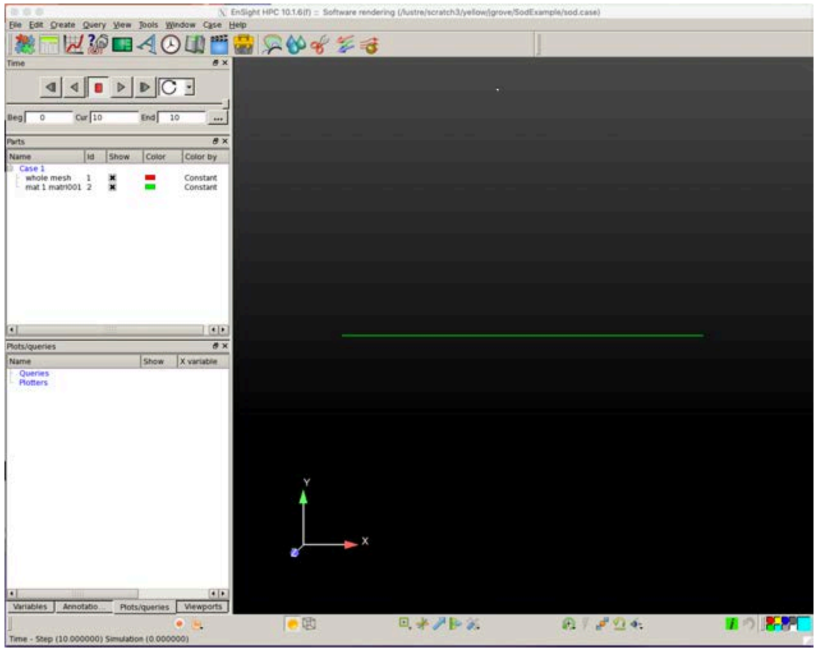
- Both Ensight and Paraview require special graphic data files to visualize a simulation
- Here we will use the EnsightGold file format supported by both Ensight and Paraview
  - There are a variety of other formats supported by the two tools
- To produce the needed graphics file you have two options
  - Enable Ensight data file printing in your input deck
  - Post process the data using AmhcTools
- To modify the input deck to produce ensight format files do the following
  - Set an input array “ensight_type” using the short names of the variables names e.g. “ensight_type(1) = 'rho', 'prs', 'tev', 'sie', ‘snd ’”
  - Set the input variable ensight_num to equal the number of fields to be printed (in this case 5)
    - When you run xrage a sudirectory called ensight will be created in the run direction with one directory for each dump printed
      - Technically this directories contain Ensight6 format files but this format is very similar to the newer EnsightGold format and the differences are not important for our current purpose.
    - Other options are available to control the frequency of printing and location of the data files
  - Each subdirectory contains image information for one time slice of the solution

Example: Using Ensight

- The other option to generate EnsightGold format files is to postprocess the restart dumps using the ReadAmhcDmp tool from the AmhcTools package
  - On HPC machines the binary is located in /usr/projects/eap/tools/amhctools/Latest/bin/ReadAmhcDmp
- Run ReadAmhcDmp using the options
  - ReadAmhcDmp -ensight EnsDir sod-dmp*
    - creates multiple subdirectories using the base name EnsDir (this is just a name you can give it any other name you like) the directory names are formed by adding an index starting at zero to the base directory name.
- The command MakeCaseFile located in /usr/projects/eap/tools/amhctools/Latest/bin/MakeCaseFile will combine all of the data from the separate directories into a single header file that will allow easy animation of the multiple time slices
  - MakeCaseFile EnsDir
    - The argument is the same name you used with ReadAmhcDmp
- MakeCaseFile doesn’t work with the Ensight6 format described in the previous slide, but it is a fairly easy edit to create a common case file.
  - The basic issue is that a combined case files wants the directories named consecutively, not by cycle number as xRage creates them

Example: Using Ensight

- I will illustrate the use of Ensight to extract plots from the Sod example using the multiple file case file created by MakeCaseFile
  - For the xRage Ensight files the same step applies but would be executed for a single frame within the appropriate ensight subdirectory
- Open ensight – “ensight101 –X sod.case”
- The window on the right will appear
- The solid line is actually a 1D constant color plot of the two parts displayed on the left hand side of the screen.
- You can turn off printing of this line by selecting the “x” next to the part name under the Show column


EnSight HPC 10.1.6(f) :: Software rendering (/lustre/scratch3/yellow/jgrove/SodExample/sod.case)

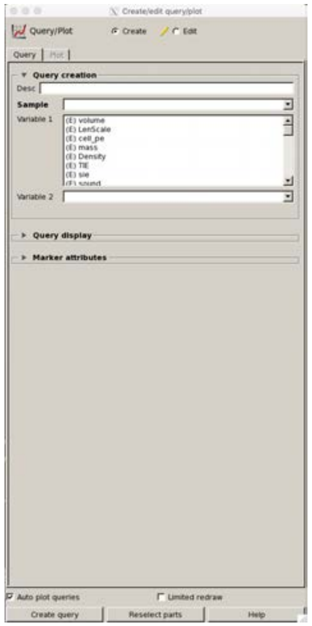
Beg 0 Cur 10 End 10

Name	Id	Show	Color	Color by
Case 1				
whole mesh	1	✖	🟥	Constant
mat 1 matri001	2	✖	🟩	Constant

Time - Step (10.000000) Simulation (0.000000)

Example: Using Ensight

- To plot a one dimensional query select the query icon near the top of the screen
- The query window on the right appears
- In the main window make sure the part "whole mesh" is selected
- In the query window under "Sample" select "At 1D part over distance"
- Select "(E) Density" from the Variable 1 list
- At the bottom of the Query/Plot window select "Create query"
- A graph of the solution density appears in the main window
- There are numerous options to modify the plot
- You can move forward and backwards in time using the animation arrows under the "Time" block
- Quit by selecting Quit under the file menu at the top of the main window
- Comprehensive documentation is available under the Help menu



Create/edit query/plot

Query/Plot [✓ Create] Edit

Query | Plot

Query creation

Desc:

Sample:

Variable 1: (E) volume, (E) LenScale, (E) cell_pe, (E) mass, (E) Density, (E) TIE, (E) sie, (E) sound

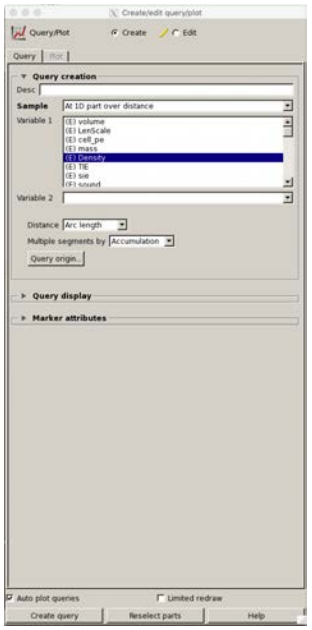
Variable 2:

Query display

Marker attributes

[✓ Auto plot queries] Limited redraw

Create query | Reselect parts | Help



Create/edit query/plot

Query/Plot [✓ Create] Edit

Query | Plot

Query creation

Desc:

Sample: At 1D part over distance

Variable 1: (E) volume, (E) LenScale, (E) cell_pe, (E) mass, [✓ (E) Density], (E) TIE, (E) sie, (E) sound

Variable 2:

Distance: Arc length

Multiple segments by: Accumulation

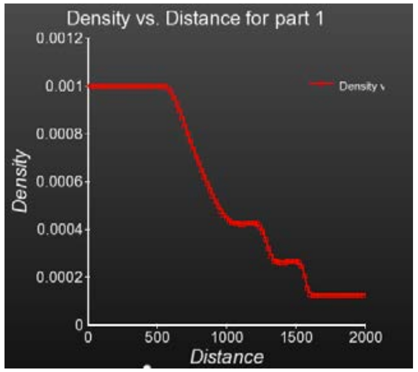
Query origin...

Query display

Marker attributes

[✓ Auto plot queries] Limited redraw

Create query | Reselect parts | Help

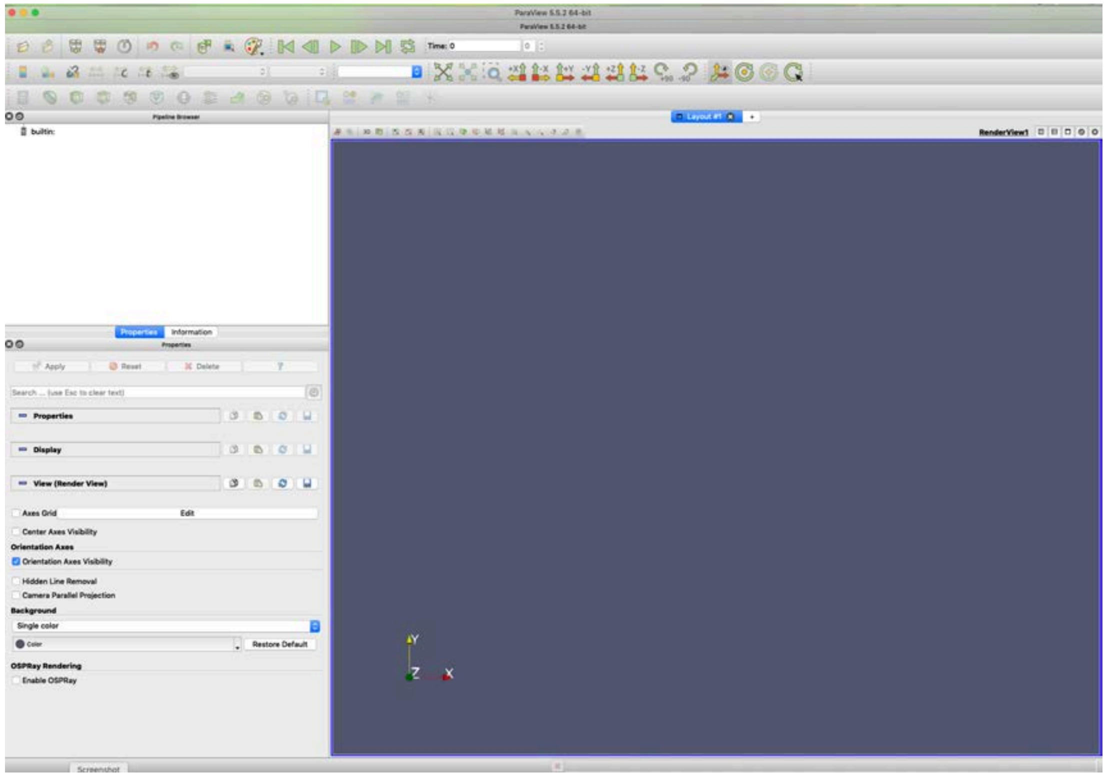


Example: Using Paraview

- Installing paraview on your client workstation requires a bit of work
  - Contact your local admin for help
  - Make sure the version of paraview agrees with the one on the server
- Configuring paraview for use
  - See https://hpc.lanl.gov/paraview_usage
    - Follow the steps outlined
    - I used dev for the quality of server and standard for the slurm partition
    - check the OS entries and the path to xterm
    - log files are in your home directory on the remote server in the directory .paraview
  - You can also use paraview in standalone mode on a workstation if the data files are available there
- Connection requires a back end node allocation. The scripts described in the above link will take care of this process, but the connection might take some time to establish
- Once connected proceed as follows
  - We will use the EnsightGold format files created for the Ensight example
    - This is one of many file formats paraview supports
    - You can generate VTK output using ReadAmhcDmp. Use the –vtk option instead of -ensight

Example: Using paraview

- Open the paraview app as appropriate for your system.
- You should get a window like the below
- My examples were generated on a MacBook Pro. The Linux behavior should be similar except for the way Macs use the tool bar.

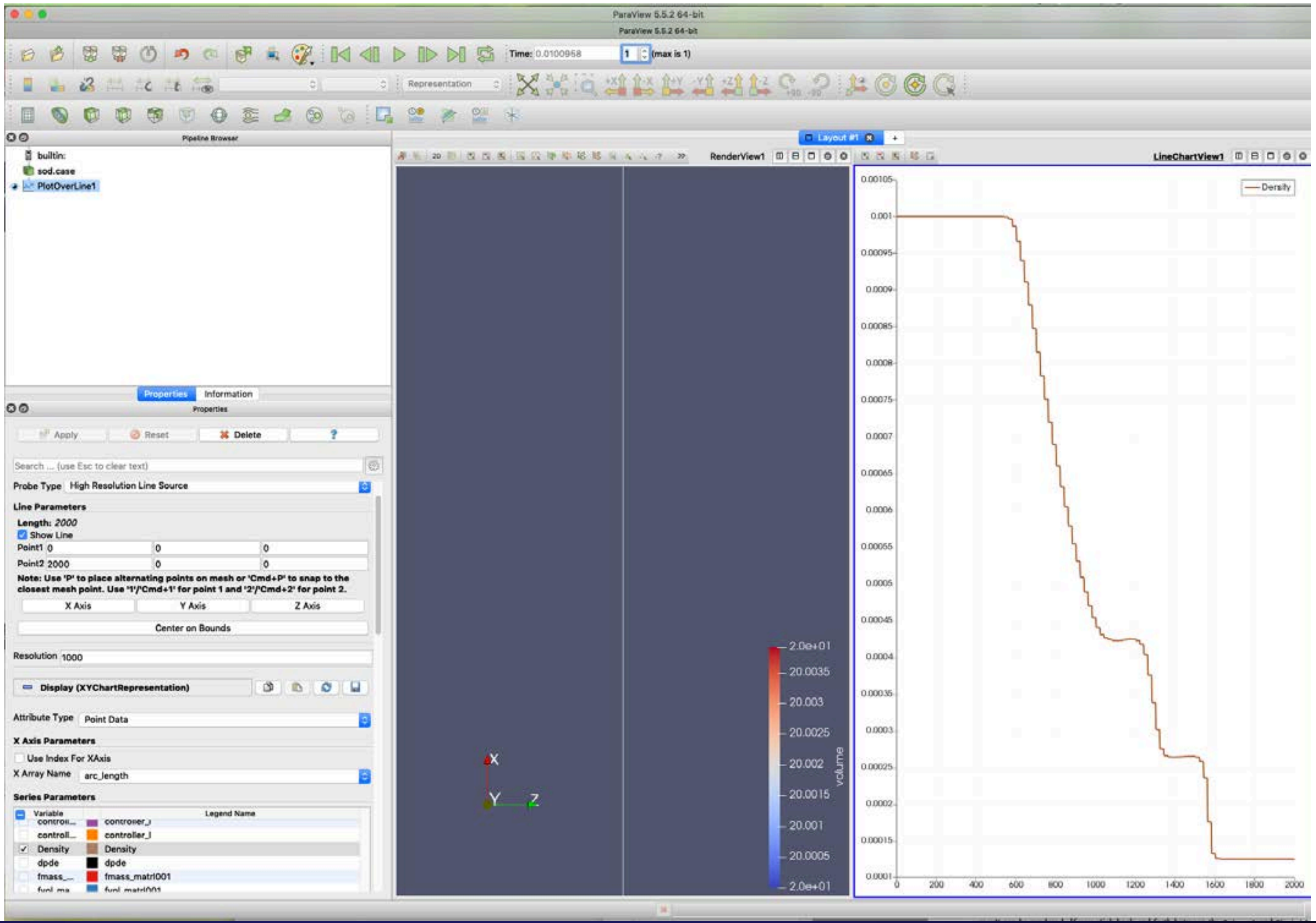


Example: Using paraview

- Select the folder option on the upper/left of the paraview window and navigate to the file you want to open.
  - Alternatively you can use the file option on the menu bar to get the same option
- In this example we select the sod.case file as we did when using Ensight
- Select Apply which will be highlighted in blue on the left side of the screen
- Go the filters tab on the menu bar and under “Data Analysis” select “Plot over Line”
- This plots everything. Scroll down under Properties until you see the variables list. I unselect everything with the blue – at the top of the list and then select only Density.
- This shows the Density field at time 0, there are two times in the plot.
- At the top of the screen You should see a Time option. Click the up arrow to get the next time.

Example: Using paraview

- If all goes well this is what you should see
- Like all powerful packages the learning curve is steep. See the user manual and ask your friends more help



Search ... (use Esc to clear text)

Probe Type: High Resolution Line Source

Line Parameters

Length: *2000*

☑ Show Line

Point1	0	0	0
Point2	2000	0	0

Note: Use 'P' to place alternating points on mesh or 'Cmd+P' to snap to the closest mesh point. Use '1'/'Cmd+1' for point 1 and '2'/'Cmd+2' for point 2.

Resolution: 1000

Display (XYChartRepresentation)

Attribute Type: Point Data

X Axis Parameters

☐ Use Index For XAxis

X Array Name: arc_length

Series Parameters

	Variable		Legend Name
	controll...	🟪	controller_i
	controll...	🟧	controller_l
✓	Density	🟫	Density
	dpde	⬛	dpde
	fmass_...	🟥	fmass_matrl001
	fvol_ma	🟦	fvol_matrl001

Advanced Input File Usage

- The full set of capabilities provided by the xRage input setup is far beyond what can be presented in a lecture
  - Literally months worth of material
- The next series of slides are only an attempt to give highlights of some of the most popular and useful tools for input file creation
- Users should make full usage of the xrage user's manual of more complete documentation
  - /usr/projects/eap/doc/xrage_userman.pdf
- As always users should make use of collogues for help and make frequent use of the crestone_support@lanl.gov email list
- In practice it is very common for people to start from input decks provided by others that are for "similar" problems to the one of interest to that person
  - The key here is to be able to read an input deck and understand what it is doing and how to modify it in a suitable way

The Parser

- The xRage Parser is a very handy tool for creating input decks
- Some of the provisions enabled by the Parser include
  - Ways to replace unmeaningful numbers by symbolic names
  - A calculator to compute expressions
    - The calculator implements the standard arithmetic operations as well as the standard transcendental functions like logarithms, exponentials, and trigonometric routines
  - Conditionals to change the behavior of an input deck
  - A do loop capability to simplify input
  - Subroutines for more complex operations
  - File inclusion
  - A mechanism to support command line arguments
  - Debugging tools to list the current state of variables and functions set up by the Parser at a given point in the input deck.

Parser Example – Physical units

- In many cases users tend to think in terms of physical units other than the cgs+electronvolt units used in xRage
  - For example temperature units of Kelvin or pressure units of megabars
- The file to the right shows a parser include file units.cgs that defines conversion factors between various unit sets
- Use of this units file can provide at least two purposes
  - Document the physical units used in a problem setup
  - Allow the use of more convenient physical units for input
    - Note the CGS unit for pressure is a microbar while many applications work in higher pressure regimes
      - A microbar is effectively a vacuum

```
! unit conversion factors for cgs units
! length
$cm = 1.0                    ! cm -> cm
$m = (100 * $cm)             ! m -> cm
! time
$sec = 1.0                   ! sec -> sec
! Mass
$g = 1.0                     ! gram -> gram
$kg = (1000 * $g)
! Temperature
$eV_k = 1.0                  ! eV/k_boltzmann -> eV/k_boltzmann
$K = (1/11604.506)           ! K -> ev/k_boltzmann
! Force
$N = ($kg*$m/$sec**2)        ! Newton (N) equals kg m / sec**2
$kN = (1.0e3 * $N)           ! kiloNewton (kN) equals 1000 Newtons
```

Example: Modified sod.input to use the Parser

```
pname = "sod"
include units.cgs

kread = -1       ! this value creates a new run as compared to a restarted run
ncmax = -1       ! no maximum cycle number, run until tmax is reached
tmax = (0.01 * $sec) ! run until simulation time tmax [seconds]
tedit = (0.01 * $sec)! output a restart dump file every tedit [seconds]

$xzero = (-10.0 * $m)
xzero = $xzero
imxset = 50        ! 50 zones starting at x = 0 as specified in http://www.thevisualroom.com/sods_test_problem.html
dxset = (0.4 * $m) ! zone size of 0.4 [m], domain runs from xzero to imxset*dxset + xzero = [-10 m to 10 m]

! MATERIALS
$P0 = (100.0 * $kN / $m**2) ! = 1 bar = 100 kN/m**2
$T0 = (0.025 * $eV_k)       ! = 290.11298 K
$T0K = ($T0 / $K)           ! T0 in degrees Kelvin
$rho0 = (1.0 * $kg / $m**3)
$gamma = 1.4
keos = 0                      ! Perfect gas equation of state
nummat = 1                    ! Only one EOS model
$Air = 1                      ! Give this material a symbolic name
matdef(16,$Air) = ($gamma - 1.0) ! Gruneisen export, perfect gas gamma = 1.4
matdef(30,$Air) = (($P0/$rho0)/(($gamma-1)*$T0)) ! CV, specific heat at constant volume, ergs/(gm eV/k)

! REGIONS
$numreg = 0 ! Start a counter to automatically count regions

$numreg = ($numreg+1) ! increment the region number counter
$RightReg = $numreg
$PR = (0.1 * $P0)
$rhoR = (0.125 * $kg / $m**3)
$sieR = (($PR/$rhoR)/($gamma -1))
matreg($RightReg) = $Air    ! Region one is the whole domain by default
rhoreg($RightReg) = $rhoR   ! Mass Density [grams/cc]
siereg($RightReg) = $sieR   ! Specific internal energy [erg/gram] pressure = (gamma-1)*rho*sie = .1 bar = 1e5 mubar

$numreg = ($numreg+1) ! increment the region number counter
$LeftReg = $numreg
$PL = $P0
$rhoL = $rho0
$sieL = (($PL/$rhoL)/($gamma - 1.0))
matreg($LeftReg) = $Air       ! Region two will overwrite region one for xlreg(2)<x<xrreg(2)
xlreg($LeftReg) = $xzero      ! Region two starts at xlreg(2) = $xzero = -10 m
xrreg($LeftReg) = (0.0 * $m)  ! Region two ends at xrreg(2) = 0 m, Initial location of the discontinuity
rhoreg($LeftReg) = $rhoL        ! Mass Density [grams/cc]
siereg($LeftReg) = $sieL        ! Specific internal energy [erg/gram] pressure = (gamma-1)*rho*sie = 1 bar = 1e6 mubar

numreg = $numreg            ! initialize $numreg regions (two in this case)
```

Explanation of the Parser modified sod.input

- Although the parser modified input is somewhat longer than the original input, it provides several enhancements
- The line "include units.cgs" enables the use of the units file previously defined. This units file can be copied to other runs and reused or installed in a central location
- All quantities now have documented physical units
- The input file now exactly copies the setup described in http://www.thevisualroom.com/sods_test_problem.html
  - In the earlier example the grid resolution was increased from 50 to 100 zones and the domain was shifted to begin at zero
- The introduction of pressures makes it clear how this input related to the original Sod problem specification
- Arithmetic expressions for the Parser are enclosed by parentheses
- The parser uses the Fortran like ** operator for exponents
- See xrage_userman.pdf for a more complete description of the parser

What's Left?

- This is a very limited description of the capabilities for simulation creation provided by xRage. Many things were left out for simplicity. Some additional important options include
  - Adaptive mesh control
    - mostly we set the input variables sizemat and sizebnd for each material to a desired resolution
  - Restart blocks, allows the user to modify the code behavior based on a variety of conditions.
  - Initialization of more complex physics such as
    - Material strength
    - Reactive chemistry
    - Radiation transport
    - Laser transport
    - Plasmas
  - Creation of more complex geometries
    - xRage supports a large of geometric operators that can be controlled by the input file to create very complex configurations
    - The Osito tool https://setup.lanl.gov/Codes/Osito/index.shtml provides a cad-cam capability for creating very complex geometries
- The user manual is the place to start to learn about these additional capabilities.